IN TH CLAIMS:

The text of all pending claims, (including withdrawn claims) is set forth below. Cancelled and not entered claims are indicated with claim number and status only. The claims as listed below show added text with underlining and deleted text with ~~strikethrough~~. The status of each claim is indicated with one of (original), (currently amended), (cancelled), (withdrawn), (new), (previously presented), or (not entered).

Please CANCEL claim 22 and AMEND claim 23 in accordance with the following:

1. (original) A method for ascertaining an assignability of at least one operating means in a computer system after at least one process accessing the operating means has stopped, comprising:

preparing a first state vector for the operating means before the process is put into operation,

preparing a second state vector for the operating means after the process has stopped, and

comparing the first and second state vectors for discrepancies in order to ascertain whether the stopping of the process has resulted in unassignable operating means .

2. (original) The method as claimed in claim 1, wherein the operating means is a shared operating means.

3. (original) The method as claimed in claim 1, wherein the operating means is a memory.

4. (original) The method as claimed in claim 1, wherein the process is a user process.

5. (original) The method as claimed in claim 1, wherein stopping arises as a result of unintentional termination of the process .

6. (original) The method as claimed in claim 1, wherein the two state vectors each comprise a plurality of parameters which relate to the assignability of the at least one operating means .

7. (original) The method as claimed in claim 1, wherein
the operating means is a memory, and
the state vectors are selected from the group consisting of amount of memory used, address of used memory portion, and identification of portion of memory that is available .

8. (original) The method as claimed in claim 1, wherein at least one of the first state vector and the second state vector records, using a process identifier, the process which is accessing the operating means .

9. (original) The method as claimed in claim 1, wherein the method takes into account at least one of all physical operating means and all virtual operating means .

10. (original) The method as claimed in claim 1, wherein some of the operating means are taken into account, in particular the shared operating means.

11. (original) The method as claimed in claim 1, wherein at least one of the first state vector and the second state vector is recorded by testing an operating system service .

12. (original) A method for unblocking an operating means in a computer system after termination of a process, comprising:

preparing a first state vector for the operating means before the process is put into operation,

preparing a second state vector for the operating means after the process has stopped,

comparing the first and second state vectors for discrepancies in order to ascertain whether the stopping of the process has resulted in unassignable operating means,

rebooting the process if the first and second state vectors match, and

starting at least one mechanism for unblocking the operating means if the first and second state vectors do not match.

13. (original) The method as claimed in claim 2, wherein the operating means is a memory.

14. (original) The method as claimed in claim 13, wherein the process is a user process.

15. (original) The method as claimed in claim 14, wherein stopping arises as a result of unintentional termination of the process .

16. (original) The method as claimed in claim 15, wherein the two state vectors each comprise a plurality of parameters which relate to the assignability of the at least one operating means .

17. (original) The method as claimed in claim 16, wherein

the operating means is a memory, and

the state vectors are selected from the group consisting of amount of memory used, address of used memory portion, and identification of portion of memory that is available.

18. (original) The method as claimed in claim 17, wherein at least one of the first state vector and the second state vector records, using a process identifier, the process which is accessing the operating means .

19. (original) The method as claimed in claim 18, wherein the method takes into account at least one of all physical operating means and all virtual operating means .

20. (original) The method as claimed in claim 19, wherein some of the operating means are taken into account, in particular the shared operating means.

21. (original) The method as claimed in claim 20, wherein at least one of the first state vector and the second state vector is recorded by testing an operating system service.

22. (cancelled)

23. (currently amended) A method of ~~assessing~~ determining whether a computer memory is blocked after a program has ~~unintentionally~~ stopped, comprising:

preparing a first state vector for the memory before the program is put into operation,

preparing a second state vector for the memory after the program has stopped, and

comparing the first and second state vectors for discrepancies in order to ~~ascertain~~ determine whether the stopping of the ~~process~~ program has resulted in ~~an unassignable~~a blocked memory portion.

24. (original) A method as claimed in claim 23, further comprising:
restarting the process if the first and second state vectors match, and
rebooting an operating system if the first and second state vectors do not match.

25. (original) A computer readable medium storing at least one program for controlling a computer to perform a method of assessing a memory after a program has unintentionally stopped, the method comprising:

preparing a first state vector for the memory before the program is put into operation,

preparing a second state vector for the memory after the program has stopped, and

comparing the first and second state vectors for discrepancies in order to ascertain whether the stopping of the process has resulted in an unassignable memory portion.